A High-Performance I/O Framework for Accelerating DNN Model Updates Within Deep Learning Workflow

GNOSIS RESEARCH CENTER

Jie Ye [1] Jaime Cernuda [1] Bogdan Nicolae [2] Anthony Kougkas [1] Xian-He Sun [1]

[1]Illinois Institute of Technology [2]Argonne National Laboratory

ILLINOIS INSTITUTE OF TECHNOLOGY

Introduction

- In a traditional deep learning workflow,
  - **Producer (Scientific AI Application):** Train a DNN model offline with a fixed set of input data —> persist the trained model to a model repository for future use.
  - **Consumer (Inference Serving System [1] [2]):** Load the pre-trained DNN model from the model repository —> offer online inference queries for end-user applications.
- In practice, modern scientific AI applications often operate in a dynamic environment where data is constantly changing over time.
- However, **offline learning** is not an ideal choice for these applications since
  - New training data is constantly accumulating over time
  - Retraining the model from scratch takes increasingly longer
- To adapt data changes, Continuous Learning is proposed to continuously (re)-train a DNN model by using some online techniques [3]
- Continuous learning implies the continuous deployment of the DNN model, **increasing the model update frequency between producers and consumers**

Motivation

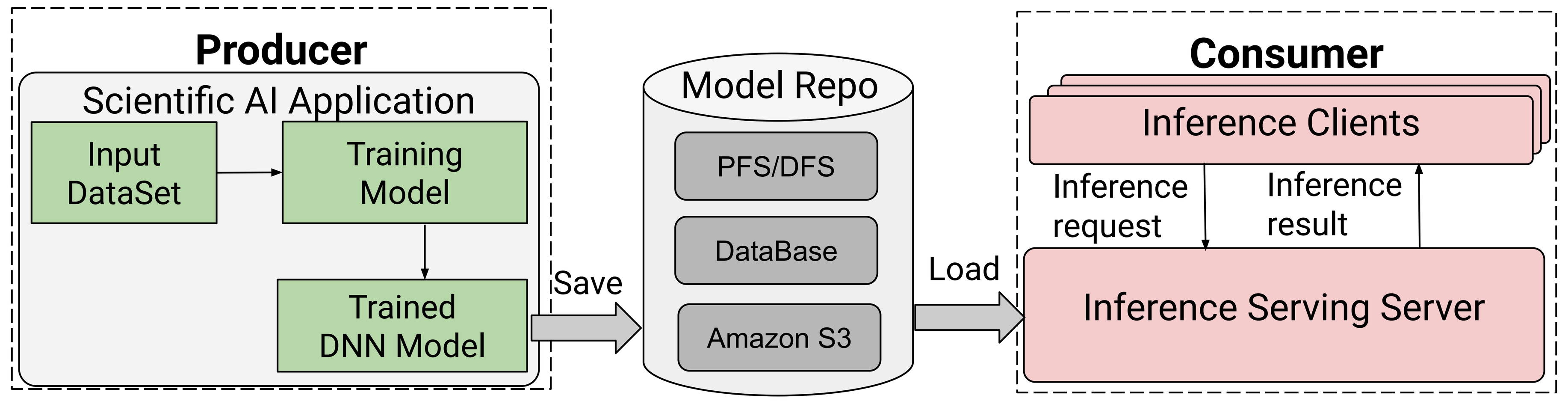

Figure 1: Coupling of training and inference within deep learning workflow

- Scientific AI Applications and Inference Serving Systems are typically coupled through a model repository (Figure 1)
  - A storage system used for sharing the trained DNN models, including PFS, DFS, DataBase, Amazon S3, Google Cloud, etc.
- Limitations
  - **High model update latency** due to the I/O bottlenecks caused by PFS and other external repository
  - **High model discovery latency on consumers** because of the static fixed-interval polling-based detection mechanism
- **Need for a direct communication channel** to speed up model updates between producers and consumers.

Challenges

- **From which source should consumers retrieve the trained model? Which data transfer method is best given the current workload?**
  - Trained models can be cached at various producer locations: GPU, DRAM, local storage, and PFS.
- **How can we choose and dynamically adjust the model update frequency to optimize both training and inference performance?**(Figure 2)
  - Frequent model updates slow down training due to continuous model checkpoint interruptions on the producer side.
  - Infrequent updates may cause inference delays or inaccurate inference results as consumers use outdated models.

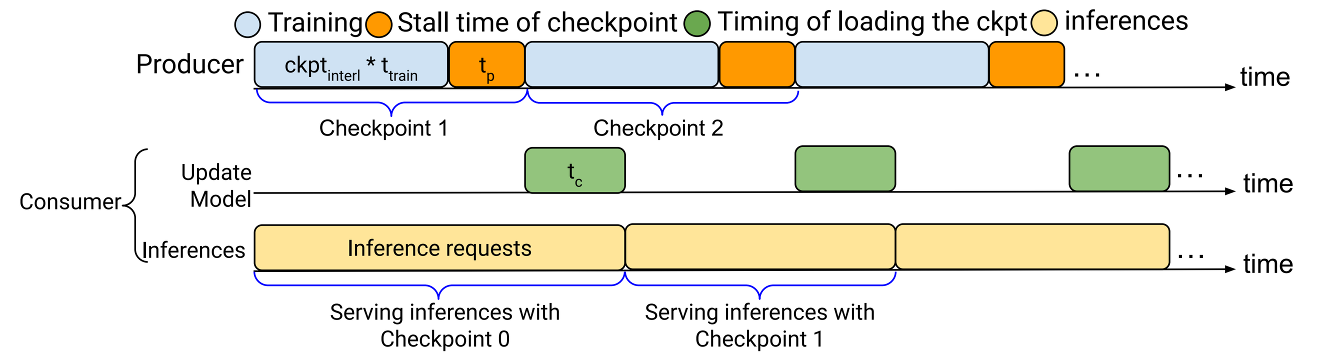

Figure 2: Impact of model update frequency to training and inference

High-Level Design

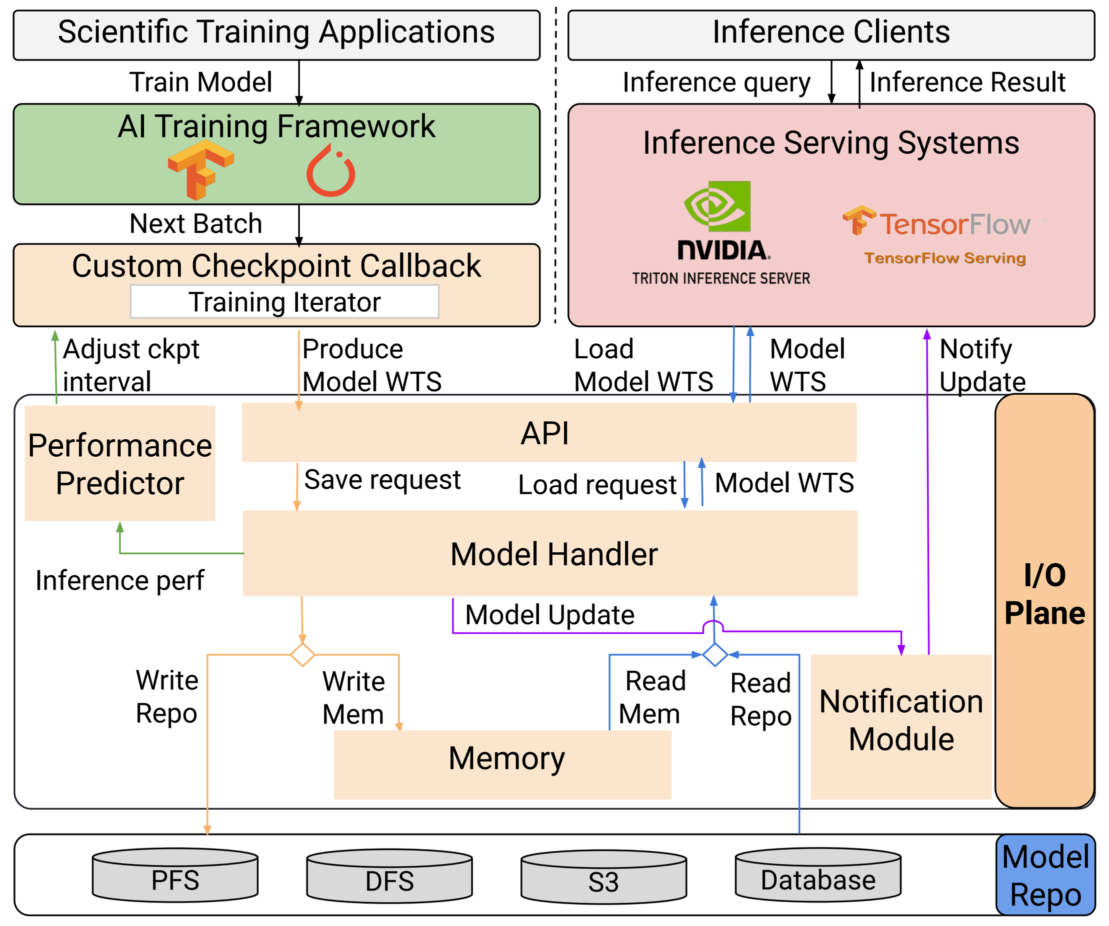

Figure 3: High-level Architecture

- This work designs and implements an I/O framework to accelerate DNN model delivery and discovery between producers and consumers (Figure 3).
- **Design Objectives**
  - **Cache-Aware Model Handler:** Reduces end-to-end model update latency.
  - **Lightweight Push-Based Notification:** Minimizes model discovery latency.
  - **Intelligent Performance Predictor:** Predict inference performance over a duration and dynamically adjust model update frequency.

Initial Evaluations

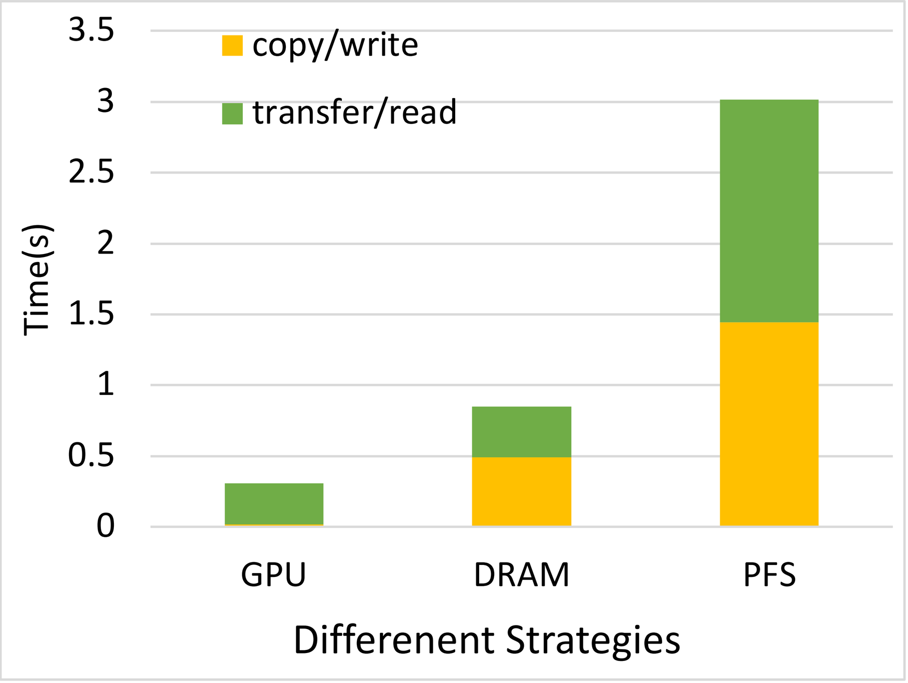

Figure 4: Latency anatomy of each stage during model update

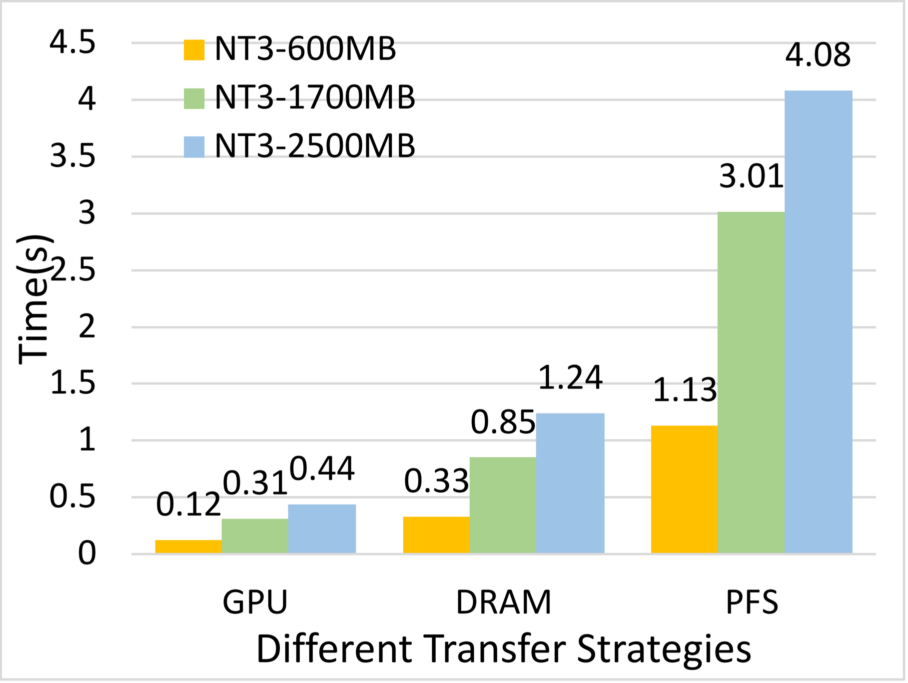

Figure 5: End-to-end model update latency when changing model size and transfer strategies

- All the tests were conducted on the Polaris [4] Supercomputer at Argonne National Laboratory
- Application: CANDLE-NT3, a representative DNN model in CANDLE (Cancer Distributed Learning Environment) project.
- **Observation — latency anatomy Of each stage during model update**
  - For GPU, the data transfer phase takes the most time, while the time of the copy phase is very mall.
  - For DRAM, the copy phase takes more time since copying a model on DRAM includes copying the tensors from GPU to DRAM and converting the tensors to NumPy arrays.
  - For PFS, the write and read latency is almost the same.
- **Observation — end-to-end model update latency analysis**
  - Compared with PFS strategy, the end-to-end model update latency is improved by 10x for GPU-to-GPU strategy and 3.5x for DRAM-to-DRAM strategy.
  - For the same data transfer strategy, the end-to-end model update latency increases linearly when increasing the model size.

Conclusions

- We designed and implemented an I/O framework to accelerate model delivery and discovery within a deep learning workflow
- Preliminary results indicate that transferring the DNN model via a direct communication channel can reduce the end-to-end latency by 10X for GPU and 3.5X for DRAM when compared to PFS.
- Future work will mainly focus on building and evaluating the performance predictor. In addition, we also need to create an algorithm that can intelligently select a data transfer strategy based on the current workloads of producers and consumers, which is a significant challenging optimization problem.

References

[1] "Tensorflow serving." https://arrow.apache.org/, 2021.

[2] "Nvidia triton inference server." https://docs.nvidia.com/deeplearning/triton-inference-server/user-guide/docs/index.html, 2023.

[3] J. Liu, B. Nicolae, D. Li, J. M. Wozniak, T. Bicer, Z. Liu, and I. Foster, "Large scale caching and streaming of training data for online deep learning," in *Proceedings of the 12th Workshop on AI and Scientific Computing at Scale using Flexible Computing Infrastructures*, pp. 19–26, 2022.

[4] "Alcf polaris supercomputer." https://www.alcf.anl.gov/polaris, 2023.

jye20@hawk.iit.edu